Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ / В.В. Шайдуров
«16» июня 2017 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ВЕДЕНИЯ ЖУРНАЛА СОБЫТИЙ ПОЛЬЗОВАТЕЛЯ

Научный руководитель
кандидат физико-математических наук,
доцент _____ / С.Н. Баранов
16.06.17

Выпускник _____ / Д.М. Иванов
16.06.17

Красноярск 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка мобильного приложения для ведения журнала событий пользователя» содержит 50 страниц текстового документа, 14 иллюстраций, 1 приложение, 13 использованных источников.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ОПЕРАЦИОННАЯ СИСТЕМА ANDROID, ЭМОЦИОНАЛЬНОЕ СОСТОЯНИЕ ЧЕЛОВЕКА, РЕЛЯЦИОННАЯ БАЗА ДАННЫХ.

Цель работы – создать мобильное приложение для ведения журнала эмоциональных состояний пользователя и составления прогноза этого состояния.

Мобильное приложение имеет дружественный интерфейс, проводит обработку данных пользователя. Языком программирования данного приложения является Java, использовалась реляционная база данных SQLite, все сделано в Android studio.

Разработанное мобильное приложение позволяет человеку оценивать свое эмоциональные состояние, позволяет повысить его стрессоустойчивость и оптимизировать свою ежедневную деятельность. Так же созданная программа может быть установлена на любые типы смартфонов, которые работают на Android, и она подстроит интерфейс под любой вид экрана.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С распространением смартфонов и планшетов всё больше людей по всему миру начинают пользоваться программами именно с мобильных устройств. Существует два направления технической реализации проектов для мобильных устройств: мобильный web сайт и мобильное приложение [4]. Мобильным web сайтом будем считать специализированный сайт, адаптированный для просмотра и функционирования на мобильном устройстве. Сайт может включать в себя интерактивные компоненты с использованием JavaScript, HTML5, новых API браузеров. В этом случае такую реализацию называют web-приложением [5]. Мобильное приложение – это специально разработанное приложение под конкретную мобильную платформу (iOS, Android, Windows Phone). Обычно приложение разрабатывается на языке высокого уровня и компилируется в нативный код ОС, дающий максимальную производительность. Существует еще третий вариант – мобильное приложение, включающее в себя компонент браузера. В этом случае часть мобильного приложения чаще всего используется для навигации и интеграции с ОС, а web-компонент – для показа контента. Обычные пользователи не могут зачастую отличить такой вариант от нативного мобильного приложения. Однако использование мобильных приложений имеет ряд преимуществ.

Главное преимущество - это привычный для пользователей смартфонов интерфейс. Ведь мобильное приложение наиболее тесно интегрировано с платформой и дает реализовать привычный отзывчивый интерфейс. Существенным в данном случае является то, что наиболее активными пользователями (теми, кто выставляет рейтинг и делает комментарии в магазинах приложений) являются те, кто предпочитает последние новшества мобильной ОС.

Следующее преимущество — это быстродействие. Web сайт, а особенно интерактивный, существенно уступает приложению с точки зрения быстродействия. Браузеры мобильных устройств пока не могут порадовать

высокой производительностью, кроме того, web-разработчики используют не самые оптимизированные версии библиотек. Однако и приложение не всегда может радовать хорошим быстродействием – излишняя анимация, сложный интерфейс значительно снижают «отклик». Кроме того, для сложной графики и анимации приходится использовать языки более низкого уровня, разрабатывать или покупать отдельные специализированные библиотеки. Эти нюансы необходимо учитывать при разработке приложений.

Очень важное преимущество мобильных приложений - интеграция с платформой. В этой области приложения далеко опережают сайт. В приложении существенно больше возможностей для доступа к устройству. Однако выше упоминался уже третий вариант, когда компонент браузера внедряется в приложение и в этом случае такая разница нивелируется. Кроме того, постоянно растет уровень предоставления доступа к возможностям устройства из браузера через расширяющийся набор API [7].

Важное отличие мобильных приложений, это относительная независимость от Интернета. Web сайт запускается из браузера, поэтому требует постоянного соединения с сетью. Это не имеет значения, если проект реализуется исключительно как онлайновый. Однако даже в этом случае из-за особенностей мобильного доступа в Интернет переход между частями приложения (навигация) связана с неприятными для пользователя задержками. Возможно, использование API для хранения локальных данных решат эту проблему, но пока примеров такого применения найти не удалось. Мобильные приложения могут осуществлять работу без подключения, выполняя кеширование и обновление данных, если требуется, при появлении соединения.

Цель работы: Создать мобильное приложение для ведения журнала эмоциональных состояний пользователя и составления прогноза этого состояния.

Понятие "эмоция" появилось в конце 19 века и связано с именами У. Джемса и Г. Ланге [3]. Проблемой диагностики эмоций занимались: А. Уэссман и Д. Рикс [1], П. Экман [8], С. В. Велиева [2].

Человек в процессе своей деятельности переживает ряд эмоций, как положительных, так и отрицательных. Эмоциональные состояния зависят от характера психической деятельности, одновременно и оказывая на нее свое влияние. При хорошем настроении активизируется познавательная и волевая деятельность человека. Эмоциональное состояние может зависеть не только от выполняемой деятельности, но и от поступка, от самочувствия, музыкального произведения, просмотренного фильма, спектакля и т.п. А самочувствие человека, в свою очередь, зависит от его эмоционального состояния. Ведь даже человек, находящийся в тяжелом состоянии, в момент эмоционального подъема, может ощутить себя совершенно здоровым.

Эмоциональные состояния преходящи, но в них отражаются индивидуальные особенности личности: у меланхолика — минорное настроение, холерика — возбужденное. Но в основном, абсолютное большинство людей при любых индивидуальных особенностях имеют усредненные, смешанные показатели активности, которое напрямую зависит от самочувствия человека и его настроения. Поэтому настроение — эмоциональное состояние, придающее окраску переживаниям и деятельности человека, оно имеет причину, не всегда осознаваемую человеком. Настроение может изменяться под впечатлением каких-либо событий, фактов, людей, окружающей природы, здоровья, выполняемой работы, учебы. В управлении настроением сказывается развитие личности.

Каждому человеку из опыта известно, насколько велика роль хорошего настроения в нашей повседневной деятельности. Веселое и серьезное, грустное и жизнерадостное — самые разнообразные настроения постоянно чередуются в нашей жизни. Настроение — это ситуативно обусловленное доминирование определенной эмоции или чувства, усиливающее или ослабляющее психическую деятельность на протяжении более или менее длительного периода. Непосредственно отражая значимость явлений, эмоции регулируют функционально-энергетический уровень жизнедеятельности. Настроение является особым видом эмоциональной регуляции жизнедеятельности в

результате непосредственного отражения связи данной ситуации с опытом человека, его установками.

Настроения, как и все другие эмоциональные состояния, являются положительными или отрицательными, они имеют определенную интенсивность, выраженность, напряженность и устойчивость. В зависимости от преобладающих в опыте данного человека эмоций и чувств соответствующее настроение становится устойчивым и характерным для данного человека. Очень важно дорожить хорошим настроением, культивировать его. Оно стимулирует нас к активной плодотворной деятельности, улучшает отношения между людьми. Человек может в известной мере регулировать свое настроение, сосредотачивая свое сознание на положительных сторонах жизни.

1 Постановка задачи

Необходимо сделать быстрый и удобный выбор текущего состояния пользователя. Например, проснулся - установил в приложении режим "проснулся", начал завтракать - установил в приложении режим "завтрак", и т.д. В ней должна быть возможность установить разные режимы: поехал на учебу, приехал на учебу, начал заниматься, начал обедать, начал заниматься спортом, начал читать книгу, начал смотреть фильм. В общем такой дневник деятельности за день. Каждые день вечером приложение просит пройти анкетирование по нескольким параметрам: усталость, сонливость, общее состояние и т.д. После того как собирается достаточная база, можно делать анализ данных, прогнозирование, рекомендации. Самое главное это то что программа должна работать на смартфоне.

2 Реализация

2.1 Android Studio

2.1.1 Описание Android Studio

Android Studio — это интегрированная среда разработки для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O.

Единая среда, в которой вы можете разрабатывать для всех Android-устройств. Основные особенности - реализована возможность вёрстки в реальном времени, доступно множество вариантов размеров и разрешений экранов. Быстрый и многофункциональный эмулятор. Мгновенный запуск для изменения изменений в рабочем приложении без создания нового APK. Присутствует раздел справки. Шаблоны кода и интеграция GitHub, чтобы помочь вам создавать общие функции приложения и импортировать пример кода. Инструменты Lint для отслеживания производительности, удобства использования, совместимости версий и других проблем. Встроены инструменты улучшения качества приложений и монетизации. Имеются инструменты для отслеживания эффективности рекламных объявлений. [10]

Встроенная поддержка облачной платформы Google, что позволяет легко интегрировать Google Cloud Messaging и App Engine. Добавлено средство взаимодействия с бета-тестерами. И многое другое.

2.1.2 Структура проекта

Файлы проекта в режиме Android. Каждый проект в Android Studio содержит один или несколько модулей с файлами исходного кода и файлами ресурсов. Типы модулей включают:

1. Модули приложений для Android
2. Библиотечные модули

3. Модули Google App Engine

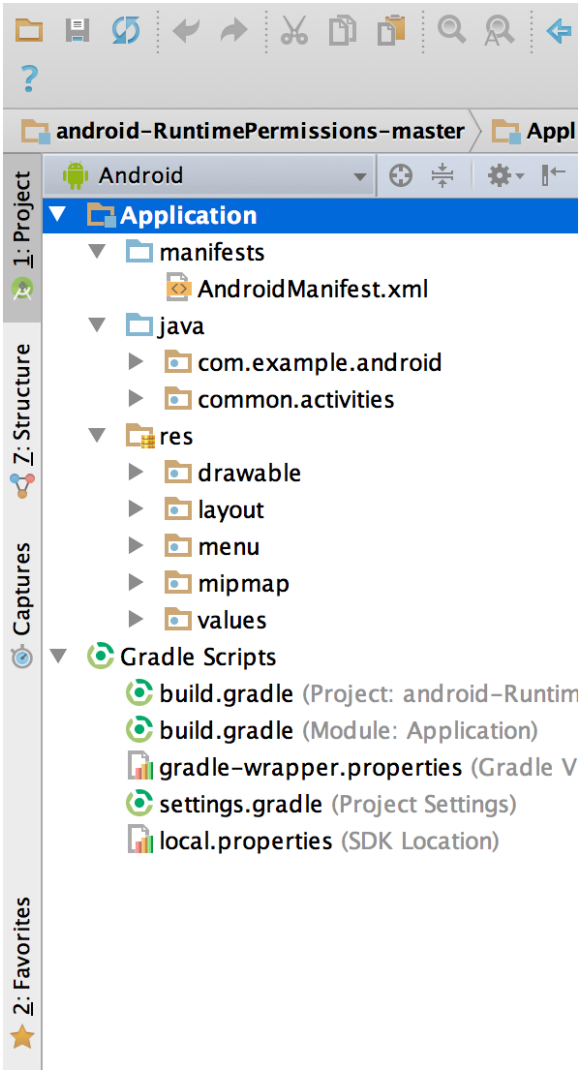

Рисунок 1 - Структура проекта

По умолчанию Android Studio отображает ваши файлы проектов в представлении проекта Android, как показано на рисунке 1. Это представление организовано модулями для быстрого доступа к исходным файлам вашего проекта. Все файлы сборки видны на верхнем уровне под Gradle Scripts, и каждый модуль приложения содержит следующие папки:

1. Manifestests: содержит файл AndroidManifest.xml.
2. Java: содержит файлы исходного кода Java, включая тестовый код JUnit.

3. Res: Содержит все ресурсы без кода, такие как XML-макеты, строки пользовательского интерфейса и растровые изображения.

Структура проекта Android на диске отличается от этого сглаженного представления [12].

2.1.3 Пользовательский интерфейс

Главное окно Android Studio состоит из нескольких логических областей, указанных на рисунке 2.

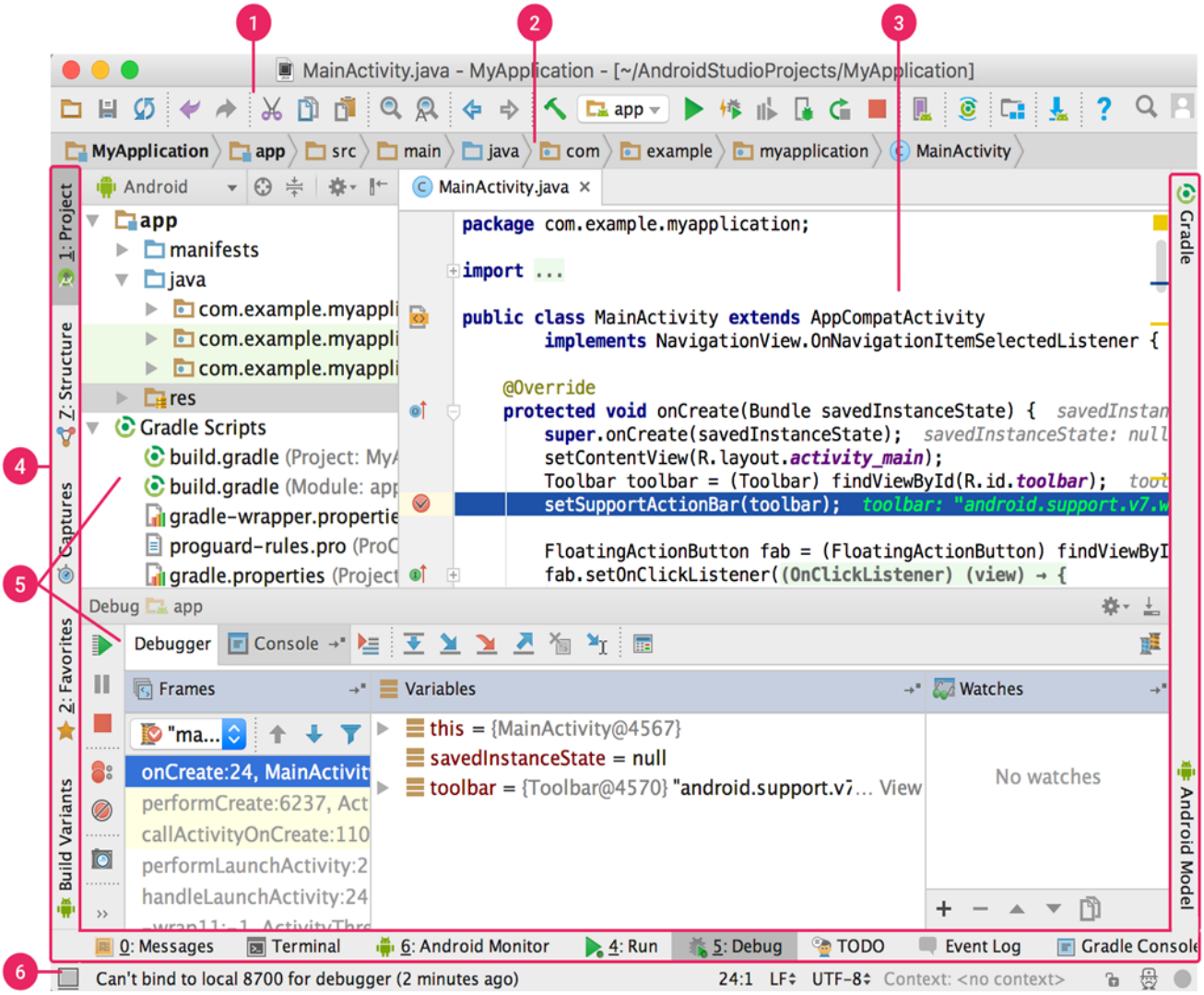

Рисунок 2 - Главное окно Android Studio

1. Панель инструментов позволяет выполнять широкий спектр действий, включая запуск приложения и запуск инструментов Android.

2. Панель навигации позволяет вам перемещаться по проекту и открывать файлы для редактирования. Он обеспечивает более компактный вид структуры, видимой в окне проекта.
3. В окне редактора вы создаете и изменяете код. В зависимости от текущего типа файла редактор может измениться. Например, при просмотре файла макета редактор отображает Редактор макетов.
4. Панель инструментов инструмента работает по внешней стороне окна IDE и содержит кнопки, которые позволяют вам развернуть или свернуть отдельные окна инструментов.
5. Окна инструментов предоставляют вам доступ к определенным задачам, таким как управление проектами, поиск, контроль версий и многое другое. Вы можете развернуть их и свернуть.
6. Строка состояния отображает статус вашего проекта и самой IDE, а также любые предупреждения или сообщения.

Вы можете организовать главное окно, чтобы предоставить себе больше места на экране, скрывая или перемещая панели инструментов и окна инструментов. Вы также можете использовать сочетания клавиш для доступа к большинству функций IDE.В любое время вы можете искать по своему исходному коду, базам данных, действиям, элементам пользовательского интерфейса и т. Д., Дважды нажав клавишу Shift или щелкнув увеличительное стекло в правом верхнем углу Android Studio окно. Это может быть очень полезно, если, например, вы пытаетесь найти определенное действие IDE, которое вы забыли, как вызвать.

2.1.4 Создание приложения

Приложения для Android пишутся на языке программирования Java. Инструменты Android SDK (Software Development Kit – комплект разработки

программного обеспечения) компилируют написанный вами код — и все требуемые файлы данных и ресурсов — в файл APK – *программный пакет Android*, который представляет собой файл архива с расширением apk. В файле APK находится все, что требуется для работы Android-приложения, и он позволяет установить приложение на любом устройстве под управлением системы Android [6].

2.2 Язык программирования Java

2.2.1 Описание языка Java

Язык программирования Java является объектно-ориентированным языком программирования, который был создан Джеймсом Гослингом и другими инженерами в компании Sun Microsystems. Он был разработана в 1991 году, как часть проекта "Green Project", и официально объявлен 23 мая 1995 года, в SunWorld, а выпущен в ноябре. Java была изначально разработана как замена для C++ (хотя набор функций больше похожа на Objective C) и известный как Дуб (в честь дерева за пределами офиса Гослинга) [9].

Существовали четыре основных цели при создании языка Java:

1. Объектно-ориентированный язык.
2. Независим от целевой платформы.
3. Должен содержать объекты и библиотеки для работы в сети.
4. Он предназначен для выполнения кода из удаленных источников надежно.

2.2.2 Объектно-ориентированный язык

Основная идея объектно-ориентированного языка в развитии программного обеспечения, все эти "вещи" (т. е. объекты) которыми можно манипулировать, а не действия, которые должны быть выполнены. Это основано на том, что первое (вещи) меняется реже и более радикально, чем действия, что

такие объекты (по сути, дела, содержащие данные) более стабильную основу для разработки программного обеспечения. Цель состоит в том, чтобы сделать крупномасштабные проекты программного обеспечения легко управляем, поэтому вы можете добиться улучшения качества и сократить количество неудачных проектов по программированию.

2.2.3 Независимость от платформы

Вторая характеристика, независимость от платформы, означает, что программы, написанные на языке Java должны работать аналогично на различном оборудовании. Программист должен быть в состоянии написать программу один раз и запустить её в любом месте. Это достигается путем компиляции Java-кода "наполовину" в байт-код - упрощенные машинные команды, которым соответствуют стандартный набор реальных команд процессору. Этот код затем необходимо запустить на виртуальной машине, то есть программе, написанной на машинном коде для взаимодействия с аппаратными средствами, которая переводит байт-код Java в пригодный для использования код на конкретном оборудовании. Кроме того, предоставляются стандартизированные библиотеки для обеспечения доступа к особенностям архитектуры конкретной машины (например, графики и сетей) единым способам. Язык Java также включает поддержку для многопоточных программ - жизненно важная необходимость для многих сетевых приложений и основа программирования.

2.3 База данных SQLite

2.3.1 Описание базы данных SQLite

SQLite — это встраиваемая кроссплатформенная база данных, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C) [11].

В моем приложении используется база данных SQLite. Я ее выбрал, потому что она доступна на любом Android-устройстве, ее не нужно устанавливать отдельно.

Всё, что вам нужно для начала работы с базой данных – это задать необходимые настройки для создания или обновления базы данных.

Так как сама база данных SQLite представляет собой файл, то по сути при работе с базой данных, вы взаимодействуете с файлом. Поэтому операции чтения и записи могут быть довольно медленными. Настоятельно рекомендуется использовать асинхронные операции, например, при помощи класса AsyncTask [13].

2.3.2 Класс ContentValues

Класс ContentValues используется для добавления новых строк в таблицу. Каждый объект этого класса представляет собой одну строку таблицы и выглядит как ассоциативный массив с именами столбцов и значениями, которые им соответствуют.

2.3.3 Курсоры

В Android запросы к базе данных возвращают объекты класса Cursor. Вместо того чтобы извлекать данные и возвращать копию значений, курсоры ссылаются на результирующий набор исходных данных. Курсоры позволяют управлять текущей позицией (строкой) в результирующем наборе данных, возвращаемом при запросе.

3 Описание программы

Интерфейс моего приложения состоит из четырех основных страниц. Общий вид приложения показан на рисунке 3.

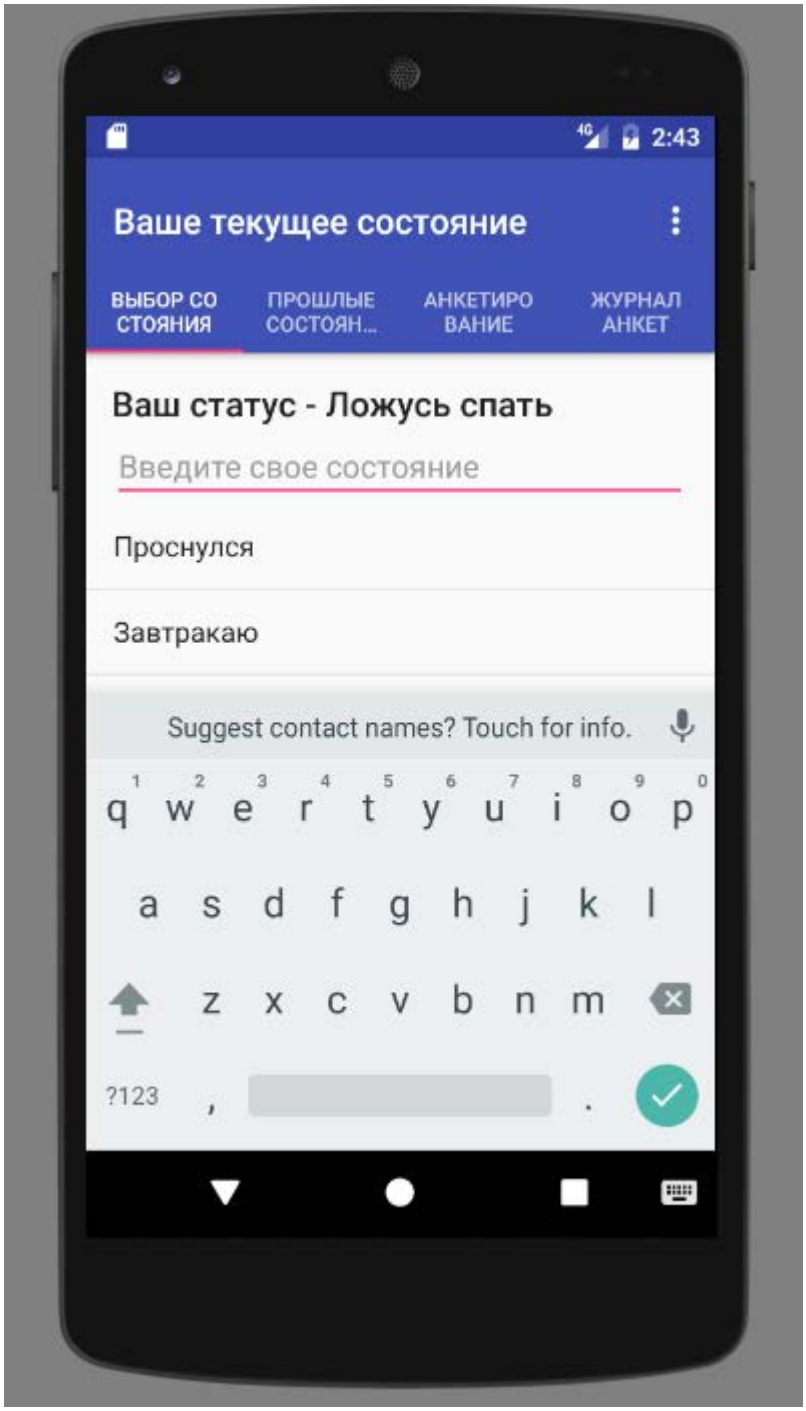

Рисунок 3 - Общий вид приложения

Рассмотрим подробнее каждый элемент приложения на экране смартфона.

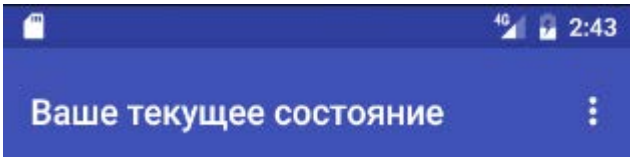

Рисунок 4 - Заголовок и дополнительное меню

В верхней части экрана приложения располагается название и дополнительное меню, как показано на рисунке 4. В дополнительном меню можно отчищать базу данных и быстро заполнять ее (для отладки приложения).

Рисунок 5 – Кнопки выбора страницы

Далее после заголовка идет выбор нужной страницы, их можно выбирать, нажимая на экран, а также можно в любом месте экрана смахивать влево и вправо чтобы перемещаться между страницами приложения (рисунок 5).

В данный момент активна страница «Выбор состояния», это видно так как она подчеркивается и выделяется ярким – белым цветом шрифта.

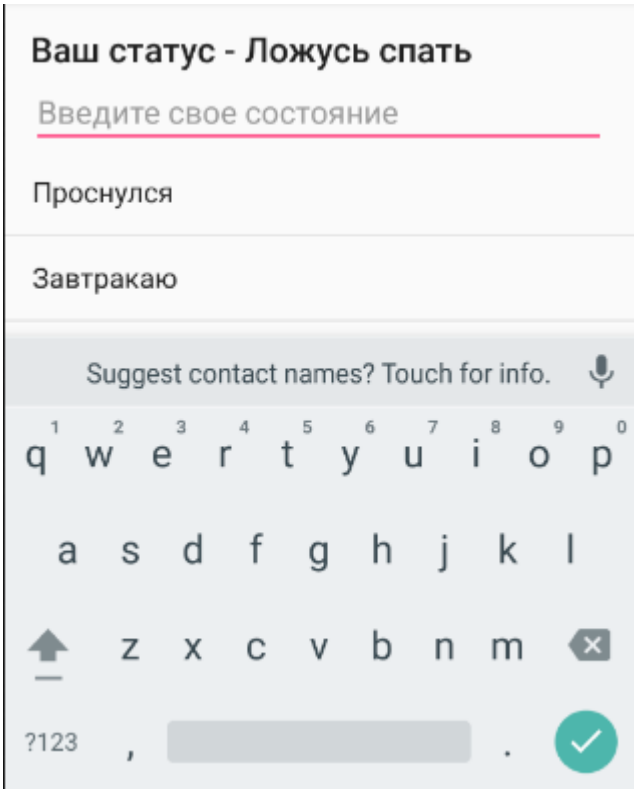

Рисунок 6 - Основной контент выбранной страницы

Далее идет основной контент выбранной страницы (рисунок 6).

3.1 Страница 1

При включении программы по умолчанию открывается страница 1, поэтому рассмотрим ее подробнее. Она состоит из трех основных элементов «TextView», «EditText» и «ListView» (рисунок 6). Рассмотрим каждый элемент подробнее.

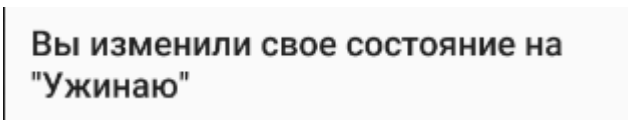

Рисунок 7 - Заголовок первой страницы

Элемент «TextView» предназначен для отображения текста без возможности редактирования его пользователем. TextView - один из самых используемых компонентов. С его помощью пользователю удобнее ориентироваться в программе. По сути, это как таблички: «Руками не трогать»,

«По газону не ходить», «Вход с собаками воспрещен», «Часы работы с 9.00 до 18.00» и т.д., и служит для представления пользователю описательного текста.

В моей программе этот элемент отвечает за заголовок страницы и показан на рисунке 7, через него показываются все подсказки и последние действия которые совершал пользователь.

Рисунок 8 - Окно ввода состояния пользователя

Элемент «EditText» это текстовое поле для пользовательского ввода, которое используется, если необходимо редактировать текст. Важно то, что «EditText» является наследником «TextView». Для быстрой разработки текстовые поля снабдили различными свойствами и дали разные имена: Plain Text, Person Name, Password, E-mail, и т. д. Я использовал свойство «Plain Text», потому что это самый простой вариант текстового поля без наворотов. В моей работе это как раз и нужно, так как всю обработку текста я провожу сам.

На рисунке 8 главное окно ввода состояния пользователя, здесь он вводит текущее свое состояние.

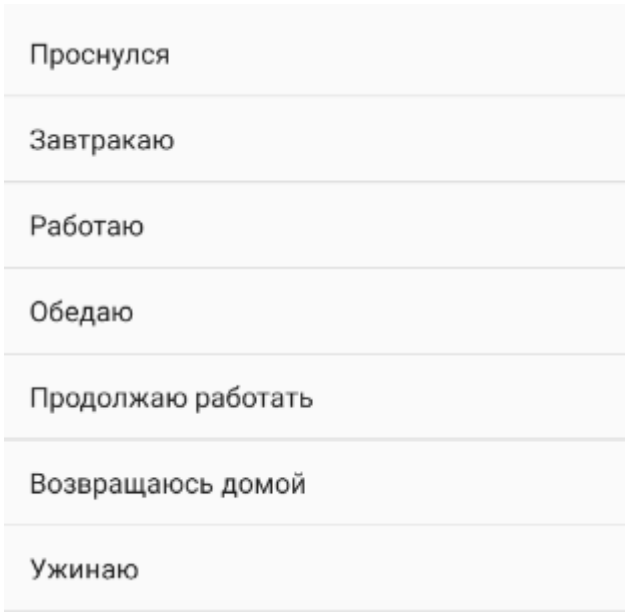

Рисунок 9 - Список популярных состояний пользователя

Элемент «ListView» представляет собой прокручиваемый список элементов. Очень популярен на мобильных устройствах из-за своего удобства. Компонент «ListView» более сложен в применении по сравнению с «TextView» и другим простыми элементами. Работа со списком состоит из двух частей. Сначала мы добавляем на форму сам «ListView», а затем заполняем его элементами списка из базы данных.

На рисунке 9 показан список самых популярных состояний пользователя для быстрого изменения состояния.

Алгоритм изменения состояния следующий:

1. Элемент «TextView» (Заголовок) изменяется на «Вы изменили свое состояние на (название состояния)»
2. Проверяется есть ли такое состояние в базе данных. Если такого состояния раньше не было в справочнике состояний, то автоматически создается новое состояние и добавляется новая кнопка с быстрым изменением состояния
3. Новое состояние сохраняется в общий журнал с датой начала действия состояния
4. Если было прошлое состояние активное, то в него записывается дата окончания действия состояния
5. Обновляется 2 страница экрана

3.2 Страница 2

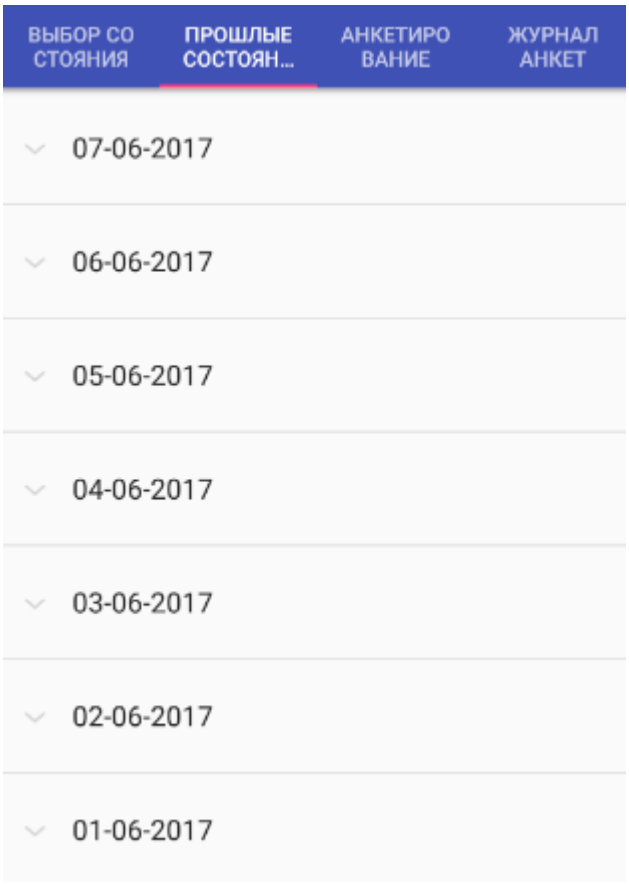

Рисунок 10 - Состояния по дням

Рассмотрим Страницу 2 подробнее (рисунок 10). Она выводит журнал прошлых состояний. Эта страница состоит всего лишь из одного элемента «ExpandableListView».

Элемент «ExpandableListView» является расширенным вариантом компонента «ListView». Основное отличие - это разворачивающий список второго уровня. Получается список в списке. Этот элемент как раз мне очень пригодился. Я смог сгруппировать все состояния по дням в основном списке (рисунок 10), и при его нажатии раскрывается подробная информация состояний о выбранном дне. На рисунке 11 пример подробной информации состояний.

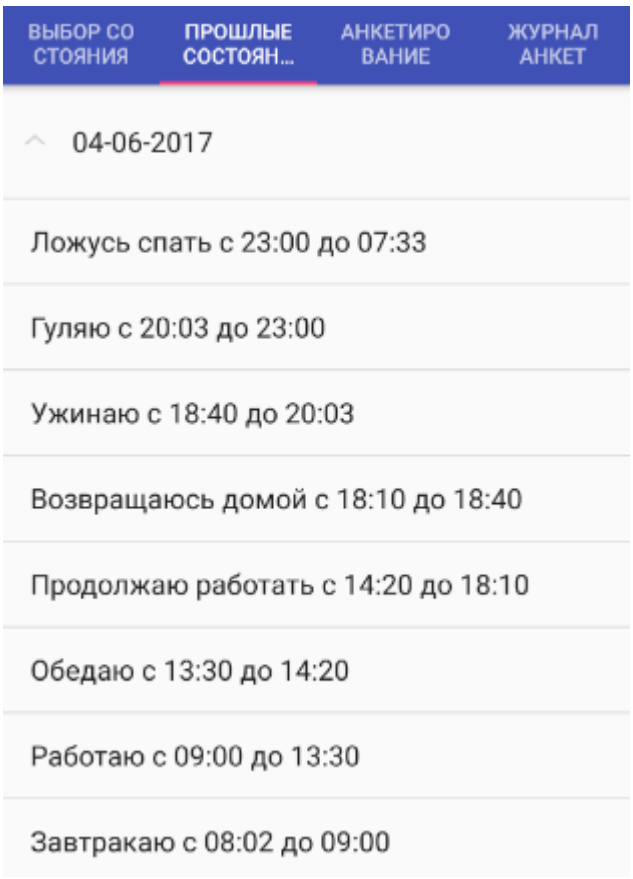

Рисунок 11 - подробная информация состояний

3.3 Страница 3

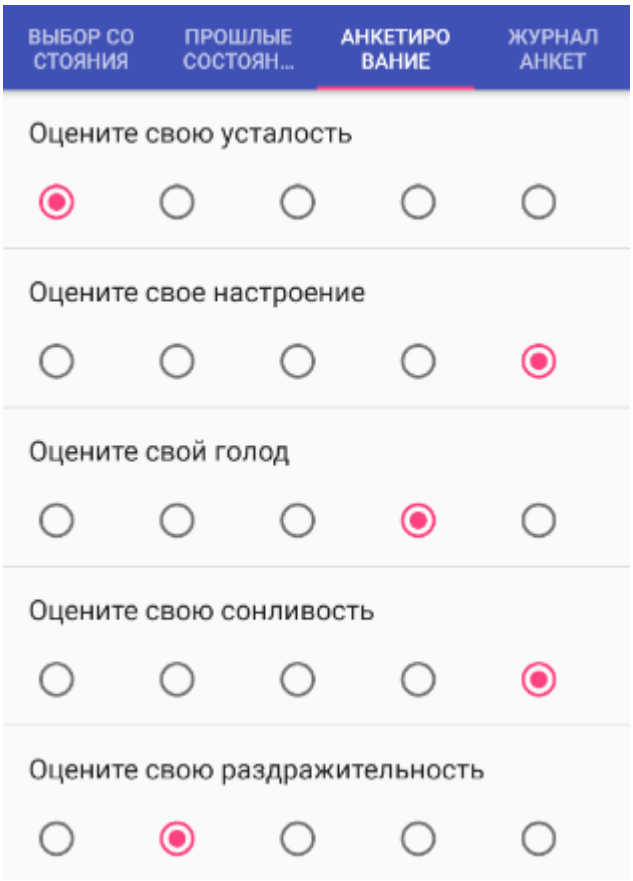

Рисунок 12 – Анкетирование

Теперь рассмотрим подробнее Страницу 3. Эта страница состоит из одного элемента «ListView». Только здесь используется не стандартная разметка для каждого элемента «ListView». При запуске приложения программа проверяет есть ли в базе данных заполненный журнал анкет на сегодня, если есть, то заполняются на этом экране все пункты анкеты. И при изменении оценки состояния обновляются прошлые состояния за текущий день. Если в базе данных нет оценок состояния на сегодня, то все кнопки оценок состояния будут не активированными.

Каждый день вечером приложение просит пройти анкетирование по нескольким параметрам: усталость, сонливость, общее состояние и т.д. и нужно оценить каждый параметр от 1 до 5 (рисунок 12). После того как собирается достаточная база данных в журнале состояний и журнале анкетирования, то

программа может делать прогнозирование эмоционального состояния в анкетировании и в будущем давать рекомендации.

3.4 Страница 4

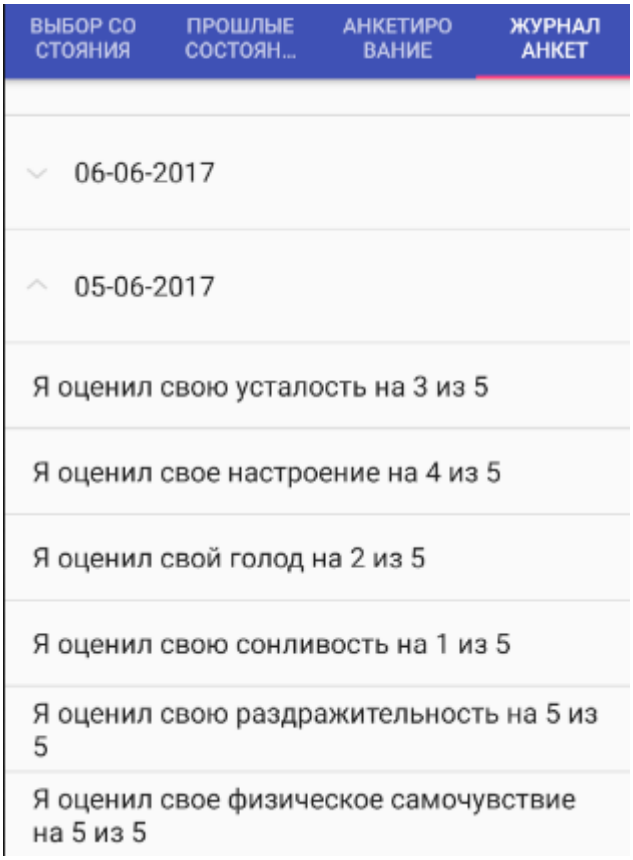

Рисунок 13

Рассмотрим последнюю страницу 4 которая показана на рисунке 13. По технической части она почти такая же, как и страница 2, только она выводит результаты анкетирования за прошлые дни, а не состояния пользователя.

3.5 Схема базы данных

В моем приложении используется 5 таблиц, ее структура показана на рисунке 14.

Подробное описание схемы тыблиц в базе данных:

1. Таблица «states» имеет 2 поля. Автоинкрементное поле «id» типа integer и поле «name» типа text.

2. Таблица «state_time» имеет 4 поля. Автоинкрементное поле «id» типа integer, поле «state_id» типа integer, поле «date» типа datetime и поле «end_time» типа datetime. Так же поле «state_id» имеет связь с таблицей «states» по «id».
3. Таблица «current_state» имеет 2 поля. Автоинкрементное поле «id» типа integer и поле «state_id» типа integer. поле «state_id» имеет связь с таблицей «state_time» по «id».
4. Таблица «questions» имеет 2 поля. Автоинкрементное поле «id» типа integer и поле «name» типа text.
5. Таблица «answer_time» имеет 4 поля. Автоинкрементное поле «id» типа integer, поле «question_id» типа integer, поле «date» типа datetime и поле «answer» типа integer. Так же поле «question _id» имеет связь с таблицей «questions» по «id».

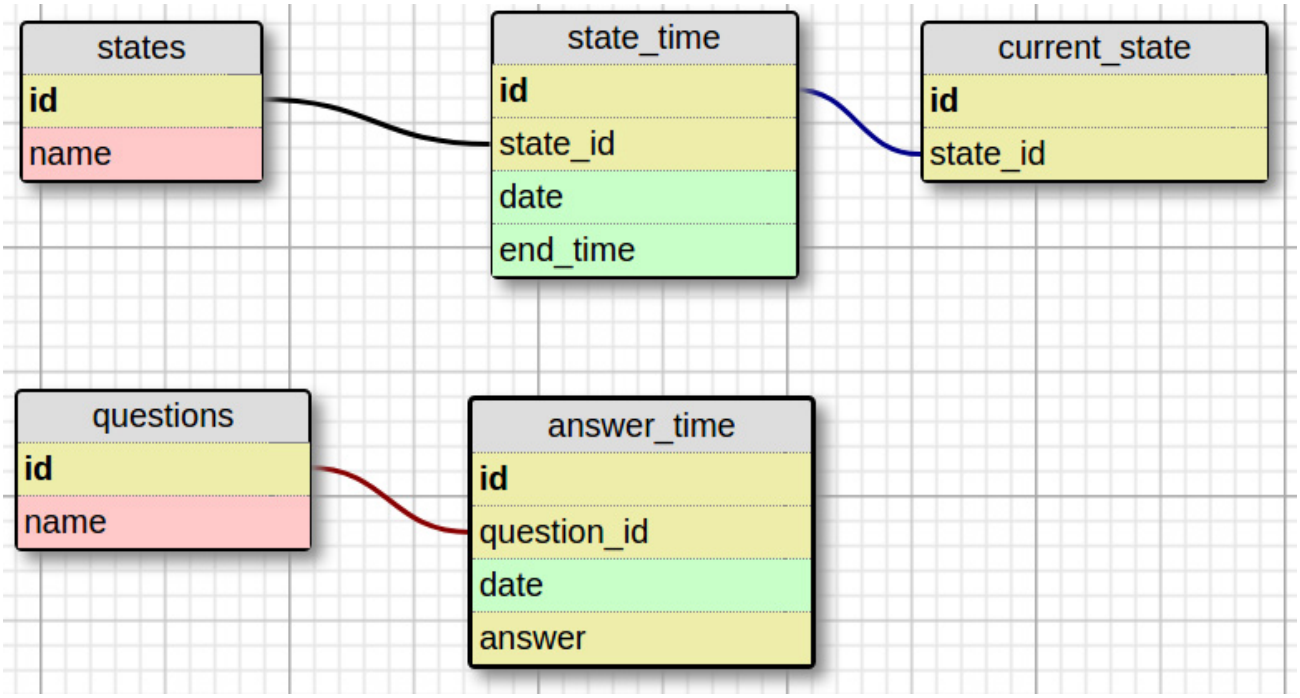

Рисунок 14 Схема базы данных

ЗАКЛЮЧЕНИЕ

Создано приложение для введения журнала пользователя. Данная программа позволяет человеку оценивать свое эмоциональные состояние, позволяет повысить его стрессоустойчивость и оптимизировать свою ежедневную деятельность. Так же созданная программа может быть установлена на любые типы смартфонов, которые работают на Android, и она подстроит интерфейс под любой вид экрана.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карелин, А. Большая энциклопедия психологических тестов / А. Карелин. – М. : Эксмо, 2007. – 416 с.
2. Велиева, С. В. Диагностика психических состояний детей дошкольного возраста / С. В. Велиева. – СПб. : Речь, 2007. – 240 с.
3. Мироненко, В. В. Хрестоматия по психологии / В. В. Мироненко. – М. : Просвещение, 1987. – 447 с.
4. Мобильный web сайт или мобильное приложение [Электронный ресурс]. – Режим доступа : https://habrahabr.ru/post/168843/, свободный. – Загл. с экрана.
5. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Н. Робин. – СПб. : Питер, 2016. – 768 с.
6. Основы создания приложений [Электронный ресурс]. – Режим доступа : https://developer.android.com/guide/components/fundamentals.html, свободный. – Загл. с экрана.
7. Харди, Б. Android. Программирование для профессионалов (2-е издание) / Б. Харди. и др. – СПб. : Питер, 2016. – 640 с.
8. Экман, П. Психология лжи. Обмани меня, если сможешь / П. Экман. – СПб. : Питер, 2010. – 304 с.
9. Язык программирования Java (Ява) [Электронный ресурс]. – Режим доступа : http://www.progaprosto.ru/doc/yazyk_programmirovaniya_java.php, свободный. – Загл. с экрана.
10. Android Studio [Электронный ресурс]. – Режим доступа : http://developer.alexanderklimov.ru/android/studio/androidstudio.php, свободный. – Загл. с экрана.
11. Meet Android Studio [Электронный ресурс]. – Режим доступа : https://developer.android.com/studio/intro/index.html, свободный. – Загл. с экрана.
12. SQLite — замечательная встраиваемая БД [Электронный ресурс]. – Режим доступа : https://habrahabr.ru/post/149356/, свободный. – Загл. с экрана.
13. SQLite на Android [Электронный ресурс]. – Режим доступа : http://developer.alexanderklimov.ru/android/sqlite/android-sqlite.php, свободный. – Загл. с экрана.

ПРИЛОЖЕНИЯ

Приложение А

Файл MainActivity.java который содержит исходный код главного класса MainActivity для запуска приложения.

```
package ru.a124au.monsgtr.states;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

  private SectionsPagerAdapter mSectionsPagerAdapter;

  private ViewPager mViewPager;

  private Page1 page1;
  private Page2 page2;
  private Page3 page3;
  private Page4 page4;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    page2 = new Page2();
    page1 = new Page1(page2);
    page4 = new Page4();
    page3 = new Page3(page4);
    mSectionsPagerAdapter                    =                    new
SectionsPagerAdapter(getSupportFragmentManager(),  page1,  page2,  page3,
page4);

    // Set up the ViewPager with the sections adapter.
```

```
    mViewPager = (ViewPager) findViewById(R.id.container);
    mViewPager.setAdapter(mSectionsPagerAdapter);

    TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
    tabLayout.setupWithViewPager(mViewPager);

  }


  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.clearDB) {
      page1.ClearDB();
      return true;
    }

    if (id == R.id.insertDB) {
      page1.InsertDB();
      return true;
    }

    if (id == R.id.insertDBAnswers) {
      page3.InsertDB();
      return true;
    }

    return super.onOptionsItemSelected(item);
  }
}
```

Файл SectionsPagerAdapter.java который содержит исходный код создания и переключения между четырьмя страницами в приложении.

```
package ru.a124au.monsgtr.states;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

public class SectionsPagerAdapter extends FragmentPagerAdapter {
```

```
  private Page1 page1;
  private Page2 page2;
  private Page3 page3;
  private Page4 page4;

  public SectionsPagerAdapter(FragmentManager fm, Page1 page1, Page2
page2, Page3 page3, Page4 page4) {
    super(fm);
    this.page1 = page1;
    this.page2 = page2;
    this.page3 = page3;
    this.page4 = page4;
  }

  @Override
  public Fragment getItem(int position) {
    switch (position) {
      case 0:
        return page1;
      case 1:
        return page2;
      case 2:
        return page3;
      case 3:
        return page4;
      default:
        return null;
    }
  }

  @Override
  public int getCount() {
    // Show 4 total pages.
    return 4;
  }

  @Override
  public CharSequence getPageTitle(int position) {
    switch (position) {
      case 0:
        return "Выбор состояния";
      case 1:
        return "Прошлые состояния";
      case 2:
        return "Анкетирование";
      case 3:
        return "Журнал анкет";
    }
    return null;
  }
}
```

Файл DBHelper.java который содержит исходный код создания и обновления базы данных SQLite.

```
package ru.a124au.monsgtr.states;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

  private static final int DB_VERSION = 6;

  final String LOG_TAG = "myLogs";

  public DBHelper(Context context) {
    // конструктор суперкласса
    super(context, "myDB", null, DB_VERSION);
  }

  @Override
  public void onCreate(SQLiteDatabase db) {

    db.execSQL("create table states ("
        + "id integer primary key autoincrement,"
        + "name text);");

    db.execSQL("create table state_time ("
        + "id integer primary key autoincrement,"
        + "state_id integer, date datetime, end_time datetime);");


    db.execSQL("create table current_state ("
        + "id integer primary key autoincrement,"
        + "state_id integer);");
    db.execSQL("INSERT INTO current_state (state_id) VALUES ('-1');");


    db.execSQL("create table questions ("
        + "id integer primary key autoincrement,"
        + "name text);");
    db.execSQL("INSERT INTO questions (id, name) VALUES (1, 'свою
усталость');");
    db.execSQL("INSERT INTO questions (id, name) VALUES (2, 'свое
настроение');");
    db.execSQL("INSERT INTO questions (id, name) VALUES (3, 'свой
голод');");
```

```
    db.execSQL("INSERT INTO questions (id, name) VALUES (4, 'свою
сонливость');");
    db.execSQL("INSERT INTO questions (id, name) VALUES (5, 'свою
раздражительность');");
    db.execSQL("INSERT INTO questions (id, name) VALUES (6, 'свое
физическое самочувствие');");


    db.execSQL("create table answer_time ("
        + "id integer primary key autoincrement,"
        + "question_id integer, date datetime, answer integer);");
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

    if (oldVersion < 2) {
      db.beginTransaction();
      try {

        db.execSQL("alter table mytable rename to states;");

        // создаем таблицу должностей
        db.execSQL("create table state_time ("
            + "id integer primary key autoincrement,"
            + "state_id integer, date datetime);");


        db.execSQL("create table current_state ("
            + "id integer primary key autoincrement,"
            + "state_id integer);");
        db.execSQL("INSERT INTO current_state (state_id) VALUES ('-
1');");


        db.execSQL("create table questions ("
            + "id integer primary key autoincrement,"
            + "name text);");

        db.execSQL("create table answers ("
            + "id integer primary key autoincrement,"
            + "question_id integer, answer integer);");

        db.setTransactionSuccessful();
      } finally {
        db.endTransaction();
      }
    }
    if (oldVersion < 3) {
      db.beginTransaction();
```

```
      try {

        db.execSQL("create table answers ("
            + "id integer primary key autoincrement,"
            + "question_id integer, answer integer);");
        db.execSQL("INSERT INTO questions (id, name) VALUES (1, 'свою
усталость');");
        db.execSQL("INSERT INTO questions (id, name) VALUES (2, 'свое
настроение');");
        db.execSQL("INSERT INTO questions (id, name) VALUES (3, 'свой
голод');");
        db.execSQL("INSERT INTO questions (id, name) VALUES (4, 'свою
сонливость');");

        db.setTransactionSuccessful();
      } finally {
        db.endTransaction();
      }
    }

    if (oldVersion < 4) {
      db.beginTransaction();
      try {
        db.execSQL("INSERT INTO questions (id, name) VALUES (5, 'свою
раздражительность');");
        db.execSQL("INSERT INTO questions (id, name) VALUES (6, 'свое
физическое самочувствие');");

        db.setTransactionSuccessful();
      } finally {
        db.endTransaction();
      }
    }

    if (oldVersion < 5) {
      db.beginTransaction();
      try {
        db.execSQL("ALTER TABLE state_time ADD end_time datetime;");
        db.setTransactionSuccessful();
      } finally {
        db.endTransaction();
      }
    }

    if (oldVersion < 6) {
      db.beginTransaction();
      try {
        db.execSQL("create table answer_time ("
            + "id integer primary key autoincrement,"
            + "question_id integer, date datetime, answer integer);");
        db.setTransactionSuccessful();
```

```
      } finally {
        db.endTransaction();
      }
    }
  }
}
```

Файл Page1.java который содержит исходный код создания и управления всеми элементами на первой странице в приложении.

```
package ru.a124au.monsgtr.states;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

public class Page1 extends Fragment {

  final String LOG_TAG = "myLogs";

  EditText inputField;
  TextView titleText;
  DBHelper dbHelper;
  View rootView;
  ArrayList<String> listItems;
  ArrayAdapter<String> adapter;
  ListView mylist;
  String currId = "0";
  Page2 page2;

  public Page1(Page2 page2) {
```

```
    this.page2 = page2;
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
              Bundle savedInstanceState) {
    rootView = inflater.inflate(R.layout.fragment_page_1, container,
false);
    inputField = (EditText) rootView.findViewById(R.id.editText);
    titleText = (TextView) rootView.findViewById(R.id.textView);
    // создаем объект для создания и управления версиями БД
    dbHelper = new DBHelper(getActivity());

    listItems = new ArrayList<String>();
    adapter = new ArrayAdapter<String>(getContext(),
        android.R.layout.simple_list_item_1, listItems);
    mylist=(ListView) rootView.findViewById(R.id.lv_page_1);
    mylist.setAdapter(adapter);
    mylist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent, View view,
                  int position, long id) {
        editState(null, id, null);
      }
    });

    inputField.setOnEditorActionListener(new
EditText.OnEditorActionListener() {
      @Override
      public boolean onEditorAction(TextView v, int actionId, KeyEvent
event) {
        if (actionId == EditorInfo.IME_ACTION_DONE) {
          String stateText = inputField.getText().toString();
          if (stateText.length() > 0) {
            editState(stateText, -1, null);
            v.setText("");
            return true;
          }
        }
        return false;
      }
    });
    LoadLastState();
    AddStates();
    return rootView;
  }

  void LoadLastState() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    String sqlQuery = "select s.name as name, st.id as id "
        + "from current_state as cs "
        + "inner join state_time as st "
```

```
        + "on cs.state_id = st.id "
        + "inner join states as s "
        + "on st.state_id = s.id";
    Cursor c = db.rawQuery(sqlQuery, null);
    if (c.moveToFirst()) {
      int nameColIndex = c.getColumnIndex("name");
      titleText.setText("Ваш статус - "+c.getString(nameColIndex));
      currId = c.getString(c.getColumnIndex("id"));
    }
    Log.d(LOG_TAG, currId);
    c.close();
    // закрываем подключение к БД
    dbHelper.close();
  }

  void AddStates() {
    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    // делаем запрос всех данных из таблицы states, получаем Cursor
    Cursor c = db.query("states", null, null, null, null, null, null);

    if (c.moveToFirst()) {
      int nameColIndex = c.getColumnIndex("name");
      do {
        listItems.add(c.getString(nameColIndex));
      } while (c.moveToNext());
    }
    adapter.notifyDataSetChanged();
    c.close();
    // закрываем подключение к БД
    dbHelper.close();
  }

  void AddStateButton(String stateText) {
    listItems.add(0,stateText);
    adapter.notifyDataSetChanged();
  }

  void editState(String stateText, long id, String time) {
    String RowID;
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    if (id != -1) {
      stateText = listItems.get((int) id);
    }
    Cursor c = db.query("states", null, "upper(name) = upper(?)", new
String[] { stateText }, null, null, null);
    titleText.setText("Вы изменили свое состояние на \"" + stateText +
"\"");
    ContentValues cv = new ContentValues();
    if (c.moveToFirst()) {
      RowID = c.getString(c.getColumnIndex("id"));
```

```
    } else {
      AddStateButton(stateText);
      cv.put("name", stateText);
      id = db.insert("states", null, cv);
      RowID = String.valueOf(id);
    }
    cv.clear();


    /**
     * Добавление нового состояния state_time
     */
    if (time == null) {
      SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      Date date = new Date();
      time = dateFormat.format(date);
      Log.d(LOG_TAG, time);
    }
    cv.put("date", time);
    cv.put("state_id", RowID);
    id = db.insert("state_time", null, cv);
    RowID = String.valueOf(id);
    cv.clear();

    /**
     * Изменение старого состояния state_time, добавление конечной метки
времени
     */
    cv.put("end_time", time);
    db.update("state_time",cv,"id = ?",new String[] { currId });
    cv.clear();
    currId = RowID;

    /**
     * Добавление current_state
     */
    cv.put("state_id",RowID);
    db.update("current_state",cv,null,null);

    /**
     * Обновление 2 экрана
     */
    page2.ReloadList();
  }

  public void ClearDB() {
    titleText.setText("Вы отчистили базу данных");
    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    // удаляем все записи
```

```
    db.delete("states", null, null);
    db.delete("state_time", null, null);
    db.delete("answer_time", null, null);
    // закрываем подключение к БД
    dbHelper.close();
  }

  public void InsertDB() {
    for (int i = 0; i < 9; i++) {
      editState("Проснулся", -1, "2017-06-0"+(i+1)+" 07:33:2"+i);
      editState("Завтракаю", -1, "2017-06-0"+(i+1)+" 08:02:1"+i);
      editState("Работаю", -1, "2017-06-0"+(i+1)+" 09:00:1"+i);
      editState("Обедаю", -1, "2017-06-0"+(i+1)+" 13:30:1"+i);
      editState("Продолжаю работать", -1, "2017-06-0"+(i+1)+"
14:20:1"+i);
      editState("Возвращаюсь домой", -1, "2017-06-0"+(i+1)+" 18:10:1"+i);
      editState("Ужинаю", -1, "2017-06-0"+(i+1)+" 18:40:1"+i);
      editState("Гуляю", -1, "2017-06-0"+(i+1)+" 20:03:1"+i);
      editState("Ложусь спать", -1, "2017-06-0"+(i+1)+" 23:00:1"+i);
    }
  }

  // вывод в лог данных из курсора
  void logCursor(Cursor c) {
    if (c != null) {
      if (c.moveToFirst()) {
        String str;
        do {
          str = "";
          for (String cn : c.getColumnNames()) {
            str = str.concat(cn + " = " +
c.getString(c.getColumnIndex(cn)) + "; ");
          }
          Log.d(LOG_TAG, str);
        } while (c.moveToNext());
      }
    } else
      Log.d(LOG_TAG, "Cursor is null");
  }
}
```

Файл Page2.java который содержит исходный код создания и управления всеми элементами на второй странице в приложении.

```
package ru.a124au.monsgtr.states;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
```

```
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ExpandableListView;
import android.widget.ListView;
import android.widget.SimpleExpandableListAdapter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class Page2 extends Fragment {

  View rootView;

  // коллекция для групп
  ArrayList<Map<String, String>> groupData;

  // коллекция для элементов одной группы
  ArrayList<Map<String, String>> childDataItem;

  // общая коллекция для коллекций элементов
  ArrayList<ArrayList<Map<String, String>>> childData;
  // в итоге получится childData = ArrayList<childDataItem>

  // список атрибутов группы или элемента
  Map<String, String> m;
  SimpleExpandableListAdapter adapter;
  ExpandableListView elvMain;
  DBHelper dbHelper;

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
              Bundle savedInstanceState) {
    rootView = inflater.inflate(R.layout.fragment_page_2, container,
false);
    dbHelper = new DBHelper(getActivity());

    //ReloadList();
    elvMain = (ExpandableListView) rootView.findViewById(R.id.elvMain);
    elvMain.setAdapter(adapter);
    ReloadList();
    return rootView;
  }

  public void ReloadList() {
    LoadList();

    // список атрибутов групп для чтения
```

```
    String groupFrom[] = new String[] {"groupName"};
    // список ID view-элементов, в которые будет помещены атрибуты групп
    int groupTo[] = new int[] {android.R.id.text1};
    // список атрибутов элементов для чтения
    String childFrom[] = new String[] {"childName"};
    // список ID view-элементов, в которые будет помещены атрибуты
элементов
    int childTo[] = new int[] {android.R.id.text1};
    adapter = new SimpleExpandableListAdapter(
        getContext(),
        groupData,
        android.R.layout.simple_expandable_list_item_1,
        groupFrom,
        groupTo,
        childData,
        android.R.layout.simple_list_item_1,
        childFrom,
        childTo);
    elvMain.setAdapter(adapter);
  }

  void LoadList() {
    // создаем коллекцию для коллекций элементов
    childData = new ArrayList<ArrayList<Map<String, String>>>();
    // заполняем коллекцию групп из массива с названиями групп
    groupData = new ArrayList<Map<String, String>>();
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    String sqlQuery = "select distinct strftime('%d-%m-%Y',date) as
dateGrupe "
        + "from state_time "
        + "ORDER BY date DESC";

    Cursor c = db.rawQuery(sqlQuery, null);

    if (c.moveToFirst()) {
      int date = c.getColumnIndex("dateGrupe");
      do {
        // заполняем список атрибутов для каждой группы (дней)
        m = new HashMap<String, String>();
        m.put("groupName", c.getString(date)); // Дата
        groupData.add(m);


        // создаем коллекцию элементов
        childDataItem = new ArrayList<Map<String, String>>();
        sqlQuery = "select s.name as name, strftime('%H:%M',st.date) as
start_time, strftime('%H:%M',st.end_time) as end_time "
            + "from state_time as st "
            + "inner join states as s "
            + "on st.state_id = s.id "
            + "where strftime('%d-%m-%Y',st.date) = ? "
```

```
            + "GROUP BY st.date ORDER BY st.date DESC";

        Cursor el = db.rawQuery(sqlQuery, new
String[]{c.getString(date)});
        if (el.moveToFirst()) {
          int   start_timeColumnIndex = el.getColumnIndex("start_time"),
              end_timeColumnIndex = el.getColumnIndex("end_time"),
              stateColumnIndex = el.getColumnIndex("name");
          do {
            String start_time = el.getString(start_timeColumnIndex),
                end_time = el.getString(end_timeColumnIndex),
                state = el.getString(stateColumnIndex),
                str = state+" с "+start_time;
            if (end_time != null) {
              str += " до "+end_time;
            }
            m = new HashMap<String, String>();
            m.put("childName", str);
            childDataItem.add(m);
          } while (el.moveToNext());
          childData.add(childDataItem);
        }
        el.close();
      } while (c.moveToNext());
    }
    c.close();
    // закрываем подключение к БД
    dbHelper.close();
  }
}
```

Файл Page3.java который содержит исходный код создания и управления всеми элементами на третьей странице в приложении.

```
package ru.a124au.monsgtr.states;


import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.ProgressBar;
```

```
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.SimpleAdapter;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

public class Page3 extends Fragment {

    View rootView;
    ArrayList<String> listItems;
    //ArrayAdapter<String> adapter;
    SimpleAdapter adapter;
    ListView mylist;
    DBHelper dbHelper;

    Page4 page4;

    final String ATTRIBUTE_NAME_TEXT = "text";
    final String ATTRIBUTE_NAME_RB = "rb";

    ArrayList<Map<String, Object>> data;

    public Page3(Page4 page4) {
        this.page4 = page4;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        rootView = inflater.inflate(R.layout.fragment_page_3, container,
false);
        dbHelper = new DBHelper(getActivity());

        // упаковываем данные в понятную для адаптера структуру
        data = new ArrayList<Map<String, Object>>();

        // массив имен атрибутов, из которых будут читаться данные
        String[] from = { ATTRIBUTE_NAME_TEXT, ATTRIBUTE_NAME_RB };
        // массив ID View-компонентов, в которые будут вставлять данные
        int[] to = { R.id.text_question, R.id.radioButtons };

        // создаем адаптер
        adapter     =     new     SimpleAdapter(getContext(),     data,
R.layout.questions,
                from, to);
```

```
        // Указываем адаптеру свой биндер
        adapter.setViewBinder(new MyViewBinder());

        // определяем список и присваиваем ему адаптер
        mylist=(ListView) rootView.findViewById(R.id.lv_page_3);
        mylist.setAdapter(adapter);
        LoadList();
        return rootView;
    }

    void LoadList() {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        String sqlQuery = "select name "
                + "from questions";
        Cursor c = db.rawQuery(sqlQuery, null);
        Random rnd = new Random();
        Map<String, Object> m;
        if (c.moveToFirst()) {
            int state_idColIndex = c.getColumnIndex("name");
            do {

                m = new HashMap<String, Object>();
                m.put(ATTRIBUTE_NAME_TEXT,                          "Оцените
"+c.getString(state_idColIndex));
                m.put(ATTRIBUTE_NAME_RB, rnd.nextInt(5)+1);
                data.add(m);

            } while (c.moveToNext());
        }
        adapter.notifyDataSetChanged();
        c.close();
        // закрываем подключение к БД
        dbHelper.close();
    }

    void editState(String stateText, long id, String time, int answer) {
        String RowID;
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        if (id != -1) {
            stateText                                                    =
data.get((int)id).get(ATTRIBUTE_NAME_TEXT).toString();
        }
        stateText = stateText.replaceAll("Оцените ", "");
        Cursor c = db.query("questions", null, "upper(name) = upper(?)",
new String[] { stateText }, null, null, null);
        ContentValues cv = new ContentValues();
        if (c.moveToFirst()) {
            RowID = c.getString(c.getColumnIndex("id"));
        } else {
            cv.put("name", stateText);
```

```
            id = db.insert("questions", null, cv); // на будущее
возможность добавлять вопросы динамически
            RowID = String.valueOf(id);
        }
        cv.clear();


        /**
         * Добавление нового состояния state_time
         */
        if (time == null) {
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd HH:mm:ss");
            Date date = new Date();
            time = dateFormat.format(date);
        }
        cv.put("date", time);
        cv.put("question_id", RowID);
        cv.put("answer", answer);
        id = db.insert("answer_time", null, cv);
        RowID = String.valueOf(id);
        cv.clear();

        page4.ReloadList();
    }

    public void InsertDB() {
        Random rnd = new Random();
        for (int i = 0; i < 9; i++) {
            editState("",     0,     "2017-06-0"+(i+1)+"     07:33:2"+i,
rnd.nextInt(5)+1);
            editState("",     1,     "2017-06-0"+(i+1)+"     08:02:1"+i,
rnd.nextInt(5)+1);
            editState("",     2,     "2017-06-0"+(i+1)+"     09:00:1"+i,
rnd.nextInt(5)+1);
            editState("",     3,     "2017-06-0"+(i+1)+"     13:30:1"+i,
rnd.nextInt(5)+1);
            editState("",     4,     "2017-06-0"+(i+1)+"     14:20:1"+i,
rnd.nextInt(5)+1);
            editState("",     5,     "2017-06-0"+(i+1)+"     18:10:1"+i,
rnd.nextInt(5)+1);
        }
    }
    class MyViewBinder implements SimpleAdapter.ViewBinder {

        @Override
        public boolean setViewValue(View view, Object data,
                                    String textRepresentation) {
            switch (view.getId()) {
                case R.id.radioButtons:
```

```
                    RadioButton        rb        =        (RadioButton)
view.findViewWithTag("rb"+data);
                    rb.setChecked(true);
                    return true;
            }
            return false;
        }
    }
}
```

Файл Page4.java который содержит исходный код создания и управления всеми элементами на четвертой странице в приложении.

```
package ru.a124au.monsgtr.states;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ExpandableListView;
import android.widget.SimpleExpandableListAdapter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class Page4 extends Fragment {


  View rootView;

  // коллекция для групп
  ArrayList<Map<String, String>> groupData;

  // коллекция для элементов одной группы
  ArrayList<Map<String, String>> childDataItem;

  // общая коллекция для коллекций элементов
  ArrayList<ArrayList<Map<String, String>>> childData;
  // в итоге получится childData = ArrayList<childDataItem>

  // список атрибутов группы или элемента
  Map<String, String> m;
  SimpleExpandableListAdapter adapter;
  ExpandableListView elvMain;
  DBHelper dbHelper;
```

```
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
              Bundle savedInstanceState) {
    rootView = inflater.inflate(R.layout.fragment_page_2, container,
false);
    dbHelper = new DBHelper(getActivity());

    //ReloadList();
    elvMain = (ExpandableListView) rootView.findViewById(R.id.elvMain);
    elvMain.setAdapter(adapter);
    ReloadList();
    return rootView;
  }

  public void ReloadList() {
    LoadList();

    // список атрибутов групп для чтения
    String groupFrom[] = new String[] {"groupName"};
    // список ID view-элементов, в которые будет помещены атрибуты групп
    int groupTo[] = new int[] {android.R.id.text1};
    // список атрибутов элементов для чтения
    String childFrom[] = new String[] {"childName"};
    // список ID view-элементов, в которые будет помещены атрибуты
элементов
    int childTo[] = new int[] {android.R.id.text1};
    adapter = new SimpleExpandableListAdapter(
        getContext(),
        groupData,
        android.R.layout.simple_expandable_list_item_1,
        groupFrom,
        groupTo,
        childData,
        android.R.layout.simple_list_item_1,
        childFrom,
        childTo);
    elvMain.setAdapter(adapter);
  }

  void LoadList() {
    // создаем коллекцию для коллекций элементов
    childData = new ArrayList<ArrayList<Map<String, String>>>();
    // заполняем коллекцию групп из массива с названиями групп
    groupData = new ArrayList<Map<String, String>>();
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    String sqlQuery = "select distinct strftime('%d-%m-%Y',date) as
dateGrupe "
        + "from answer_time "
        + "ORDER BY date DESC";
```

```
    Cursor c = db.rawQuery(sqlQuery, null);

    if (c.moveToFirst()) {
      int date = c.getColumnIndex("dateGrupe");
      do {
        // заполняем список атрибутов для каждой группы (дней)
        m = new HashMap<String, String>();
        m.put("groupName", c.getString(date)); // Дата
        groupData.add(m);


        // создаем коллекцию элементов
        childDataItem = new ArrayList<Map<String, String>>();
        sqlQuery = "select s.name as name, strftime('%H:%M',st.date) as
start_time, st.answer as answer "
            + "from answer_time as st "
            + "inner join questions as s "
            + "on st.question_id = s.id "
            + "where strftime('%d-%m-%Y',st.date) = ? "
            + "GROUP BY st.date ORDER BY st.id";

        Cursor el = db.rawQuery(sqlQuery, new
String[]{c.getString(date)});
        if (el.moveToFirst()) {
          int   start_timeColumnIndex = el.getColumnIndex("start_time"),
              answerColumnIndex = el.getColumnIndex("answer"),
              stateColumnIndex = el.getColumnIndex("name");
          do {
            String start_time = el.getString(start_timeColumnIndex),
                answer = el.getString(answerColumnIndex),
                state = el.getString(stateColumnIndex),
                str = "Я оценил "+state+" на "+answer+" из 5"; //+" в
"+start_time;
            m = new HashMap<String, String>();
            m.put("childName", str);
            childDataItem.add(m);
          } while (el.moveToNext());
          childData.add(childDataItem);
        }
        el.close();
      } while (c.moveToNext());
    }
    c.close();
    // закрываем подключение к БД
    dbHelper.close();
  }
}
```

Файл activity_main.xml который содержит исходный код расположения всех элементов и их стилей на всем экране в приложении.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/main_content"
  android:layout_width="match_parent"
android:layout_height="match_parent"
  android:fitsSystemWindows="true"
tools:context="ru.a124au.monsgtr.states.MainActivity">

  <android.support.design.widget.AppBarLayout android:id="@+id/appbar"
    android:layout_width="match_parent"
android:layout_height="wrap_content"
    android:paddingTop="@dimen/appbar_padding_top"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar android:id="@+id/toolbar"
      android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
      android:background="?attr/colorPrimary"
      app:popupTheme="@style/AppTheme.PopupOverlay">

    </android.support.v7.widget.Toolbar>

    <android.support.design.widget.TabLayout android:id="@+id/tabs"
      android:layout_width="match_parent"
android:layout_height="wrap_content" />

  </android.support.design.widget.AppBarLayout>

  <android.support.v4.view.ViewPager android:id="@+id/container"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</android.support.design.widget.CoordinatorLayout>
```

Файл fragment_page_1.xml который содержит исходный код расположения всех элементов и их стилей на первой странице в приложении.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
```

```
  <LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="15dp"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"
    android:orientation="vertical">

    <TextView
      android:text="@string/title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textView"
      android:textAppearance="@style/TextAppearance.AppCompat.Title"
      />

    <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:inputType="textPersonName"
      android:ems="10"
      android:id="@+id/editText"
      android:selectAllOnFocus="false"
      android:hint="@string/input_state"
      android:imeOptions="actionDone"
      android:paddingTop="10dp"
      android:paddingBottom="10dp" />

  </LinearLayout>

  <ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/lv_page_1"
    android:footerDividersEnabled="false"
    android:headerDividersEnabled="false" />


</LinearLayout>
```

Файл fragment_page_2.xml который содержит исходный код расположения всех элементов и их стилей на второй и четвертой страницах в приложении.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
```

```
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
  <ExpandableListView
    android:id="@+id/elvMain"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
  </ExpandableListView>
</LinearLayout>
```

Файл fragment_page_3.xml который содержит исходный код расположения всех элементов и их стилей на третьей странице в приложении.

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/lv_page_3" />

</LinearLayout>
```

Файл questions.xml который содержит исходный код расположения всех элементов вопросов и их стилей на странице анкетирования в приложении.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical">

  <TextView
    android:id="@+id/text_question"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceListItemSmall"
    android:gravity="center_vertical"
    android:minHeight="?android:attr/listPreferredItemHeightSmall"
    android:paddingLeft="15dp"
    android:paddingRight="15dp" />

  <RadioGroup
```

```
    android:id="@+id/radioButtons"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="left|center_horizontal|center"
    android:paddingBottom="10dp"
    android:paddingLeft="15dp">

    <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton1"
      android:layout_weight="1"
      android:checked="false"
      android:tag="rb1" />

    <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton2"
      android:layout_weight="1"
      android:tag="rb2" />

    <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton3"
      android:layout_weight="1"
      android:tag="rb3" />

    <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton4"
      android:layout_weight="1"
      android:tag="rb4"/>

    <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton5"
      android:layout_weight="1"
      android:tag="rb5" />
  </RadioGroup>

</LinearLayout>
```